班级 计算机203 学号 202007020625 姓名 於俊涛 报告日期 11/15

**实验题目： 栈和队列数据结构的关键算法实现及性能分析**

**一、实验目的**

**1. 能够运用高级程序设计技术实现栈和队列数据结构及其关键算法；**

**2. 具体任务参加各个实验的任务书。**

**二、实验原理**

1.栈和队列数据结构的基本原理：

* 栈：
* ADT Stack{
  + - * D={ai|ai==ElemSet,i=1,2,3,…n,n>=0}
      * R={<ai-1,ai>|ai-1,ai==D,i=2,…,n}//an为栈顶，a1为栈底//
      * }ADT Stack;
* 以顺序栈和链栈两种方法储存；
* 初始化—>入栈—>出栈—>取栈顶元素；
* 队列：
* ADT Queue{
  + - * D={ai|ai==ElemSet,i=1,2,3,…n,n>=0}
      * R={<ai-1,ai>|ai-1,ai==D,i=2,…,n}//an为队列头，a1为队列尾//
      * }ADT Queue;
* 以顺序表示和链式两种方法储存；
* 初始化—>求队列长度—>入队—>出队—>取队头元素；

2.关键算法设计原理及性能影响因素分析

#define MaxSize 100 //顺序栈初始空间分配量

Typedef struct{

SElemType \*base;//栈底指针

SElemType \*top;//栈顶指针

int stacksize;//栈可用的最大容量

}SqStack;

顺序栈虽然可以在“满员”时重新分配空间扩大容量，但是工作量较大，无法预估栈是否达到最大容量；

#define MaxSize 100 //队列可能达到的最大长度

Typedef struct{

QElemType \*base;//储存空间的基地址

Int front; //头指针

Int rear; //尾指针

}sqQueue;

由于“队尾入队，队头出队”这种受限制的操作造成“假溢出”现象，需要将顺序队列变为一个环状空间，即循环队列；

**三、实验方案设计**

1.存储方案设计

顺序栈：受最大空间容量的限制，虽然可以在“满员”时重新分配空间扩大容量，但是工作量较大，无法预估栈是否达到最大容量；

链栈：不需要附加头结点，不需要判断栈是否满；

顺序表示：“假溢出”现象，转为循环队列，但是无法预估所用队列最大长度；

链队：出队时需要考虑队列最后一个元素删除后，队尾指针也丢失了，需对队尾指针重新赋值；

2.算法的设计和实现

bool isEmpty()*/\*判栈是否为空，空返回true，非空返回false \*/*

{

if (top == -1)

return true;

else

return false;

}

void Push(int x) */\*元素x入栈\*/*

{

if (top == MaxSize - 1)

return ;

else

{

top++;

mystack[top] = x;

}

}

int getTop()*/\*取栈顶元素\*/*

{

if (top == -1)

return 0;

else

return mystack[top];

}

算法时间，空间复杂度：O(log2n)；

int QueueLen(SqQueue Q){

return (Q->rear - Q->front + MAXQSIZE ) % MAXQSIZE;

}

int EnQueue(SqQueue &Q, Person e){

Q->rear = (Q->rear + 1) %MAXQSIZE;

Q->data[Q->rear] = e;

return 0;

}

int QueueEmpty(SqQueue &Q){

if(Q->front==Q->rear){

return 1;

}else{

return 0;

}

}

int DeQueue(SqQueue &Q, Person &e){

Q->front=(Q->front+1)%MAXQSIZE;

e=Q->data[Q->front];

return 0;

}

void DancePartner(Person dancer[], int num){

for(int i=0;i<num;i++){

if(dancer[i].sex=='M'){

EnQueue(Mdancers,dancer[i]);

}else{

EnQueue(Fdancers,dancer[i]);

}

}

while(QueueEmpty(Mdancers)!=1&&QueueEmpty(Fdancers)!=1){

Person x,y;

DeQueue(Mdancers, x);

DeQueue(Fdancers, y);

cout<<y.name<<" "<<x.name<<endl;

}

}

算法时间，空间复杂度;O(n)；

3.测试方案

测试一: 23；10111；



测试二：

6

雷2 M

朱1 F

杨2 M

刘1 F

张2 M

於1 F

